Can a human or computer solve 10^120 chess games?

By Sulaiman Barzangi

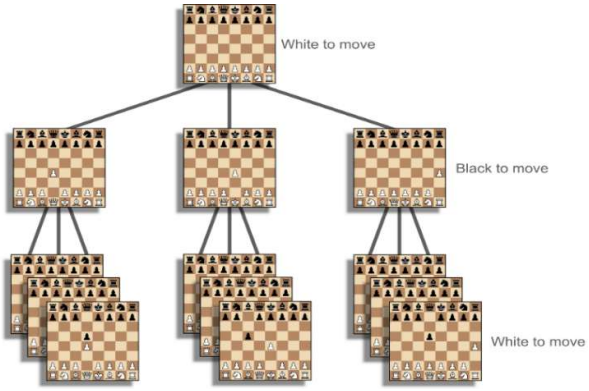
I have probably played over 10,000 chess games. Not one game has been identical to another. You'd think that if I keep playing chess, there will be a time when I can just stop thinking and just retrace my steps from that game against my cousin a few months before. But to my misfortune, my opponent doesn't play the same move my cousin did but rather catches me off guard with something new, and now I must properly think about what to play next. Would this be different if I played 100,000 games instead? Or maybe a million games? Chess does not include a roll of dice, nor is it decided by luck and randomness, it's just a set of simple rules throughout the game, and that's it. A system like that should be solved by now, just like Rubik's Cube.

In March 1950 Claude Shannon ran into the number 10^120 which approximates the number of chess games possible. Now I may love chess, but even if I had the time to play that many unique games, I would go insane before that happens (Just like Bobby Fischer). The sheer number of games possible in chess is so vast that it becomes impossible to analyze.

The real question is if chess can be considered solvable even if it cannot be computed in practice.

Imagine this: You're playing in an intergalactic chess tournament and match up in the finals against Grandmaster Xilimbljor, an intergalactic being 781 billion light years away on the planet Zuliptanius (allegedly). Xilimbljor is a 12-time back-to-back chess champion of our universe, and for a good reason too. He can compute all 10^{120} chess games in his head in an instant. You decide that he's a bit much and instead of playing against him, you file a complaint with FIDE (Intergalactic chess federation) that no organism or computer is or ever will be capable of doing that. Xilimbljor is then removed from the competition, and you win by default (I bet you he didn't expect that chess game outcome with his alien intellect). This is because chess is only solved in the hypothetical field and cannot be solved in the physical field. Which is enough proof to argue that Xilimbljor isn't from this universe and therefore does not meet the conditions to play in this tournament.

To understand why chess is physically unsolvable in our universe, consider that each position offers many moves. After each move, the pathways branch out, forming a structure known as a game tree. On average, a single position in chess has around 30 to 35 legal moves available, and after each move, a new position is formed with its own new set of possibilities. This increases the number of possible positions exponentially with each move played, causing a branching effect. For instance, after just a small number of moves, the total number of possible positions already reaches into the billions. This rapid growth in numbers makes it impractical to analyse every possible continuation, which supports the idea that although chess is deterministic, it cannot be fully solved in practice, unless you're Xilimbljor who doesn't follow the laws of this universe.



The model to identify the number of possible games was thought out by Shannon to be approximately:

$$N \approx a^{2n}$$

The branching factor a is also the number of available legal moves, the branching effect is basically multiplying the number with itself depending on the number of times it branches off n, which is the number of moves. 2n is used in the equation because each move consists of 1 ply each by both white and black, branching the game off twice per move.

When Shannon used this model to estimate the number of possible games, he assumed the value of a to be a constant between 30 and 35.

So, by using the model for N, we get the following exponential relationship:

Moves (n)	Estimated Number of Possibilities (N where $a = 35$)
1	1.23*10^3
2	1.50*10^6
3	1.84*10^9
4	2.25*10^12
5	2.76*10^15
10	7.61*10^30
15	2.10*10^46
20	5.79*10^61

Chess TMR | Desmos

After only 3 moves, the number of possible outcomes is already in the billions, and after 52 moves, that number would have surpassed 10^{80}, which is approximately the number of particles in the observable universe, according to Arthur Eddington.

Although this model of N approximates the number of games just fine, it ignores one of the fundamentals in chess: The number of legal moves available vary drastically based on the position. In the starting position there are 20 moves, but in the midgame, it could be more than 40. In fact, Nenad Petrovic discovered that the most legal moves available is 218. These different values for a are not applicable in the initial model since a is not constant, Xilimbljor, who knows all the different values of a, needs a refined model that takes different values into account.

A more refined model would be one that records all possible games, with a set value a for each position:

$$N = a_1 * a_2 * a_3 * \dots * a_{2n}$$

$$log_{10}(N) = log_{10}(a_1 * a_2 * a_3 * \dots * a_{2n})$$

$$log_{10}(N) = log_{10}(a_1) + log_{10}(a_2) + log_{10}(a_3) + \cdots + log_{10}(a_{2n})$$

$$log_{10}(N) = \sum_{i=1}^{2n} log_{10}(a_i)$$

This formulation allows for the branching factor to be implicated into the model. Although a_i differs throughout the game, for beings who aren't like Xilimbljor, the average value for a_i remains sufficient that the sum grows approximately linearly with the number of moves*:*

$$log_{10}(N) \approx 2n * log_{10}(\bar{a})$$

where $\bar{a}$ is the average branching factor (between 30-35). Since $log_{10}(\bar{a})$ is a positive constant, $log_{10}(N)$ linearly increases as n increases; therefore, N itself increases exponentially with n. This demonstrates that even a small increase in moves results in a massive increase in possible games, reinforcing the idea that complete computation is infeasible.

Because the different values for a_i varies a lot in a way where we can't plug all the values in, the refined model does not aim to produce a fundamentally different estimate to the simple model, but rather to justify the exponential model by accounting for variability in the branching factor.

If there are 10^{120} possible games, a computer could not compute that before the time Earth gets destroyed by the sun, or the heat death of the universe. For instance, if the fastest supercomputer on Earth, El Capitan, manages to analyze 10^{18} or a quintillion game possibilities a second, it would still take over $3.17 * 10^{94}$ years to fully compute. To put that into perspective, if you had started this calculation since the start of the Big Bang, you would only be about*:*

$$\frac{Time\ since\ bigbang}{El\ Capitan\ time} * 100\% \rightarrow 4.35 * 10^{-82}\% \text{ complete.}$$

Similarly, Shannon explained that solving chess with brute-force is impossible. An alternative way to solve chess, is to simplify chess itself.

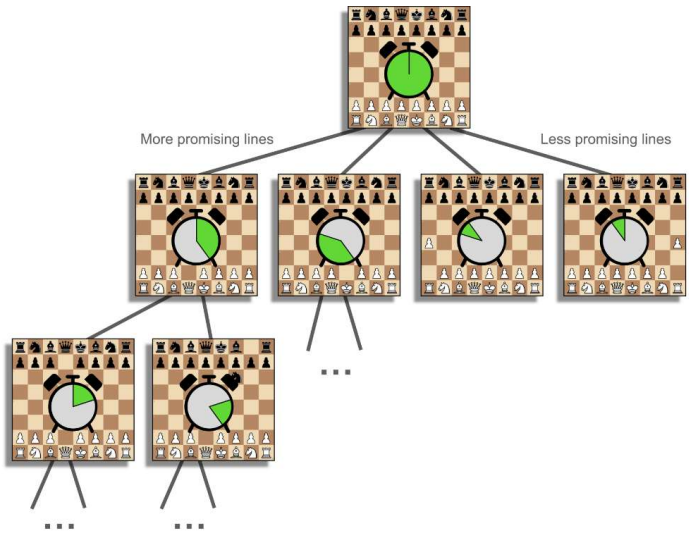
Instead of considering every game, considering only logical games is a more feasible approach, which is what chess engines do to generate the best moves. Instead of branching the game tree into ~35 branches each turn, only consider the best moves available, this is called a Heuristic Evaluation. Shannon used a heuristic evaluation based on an evaluation function that estimates a position for optimal long-term advantages based on material advantage, pawn formation, position of pieces, piece activity, and mobility.

This is paired with a Minimax algorithm which explores possible moves ahead of the game tree and scores the position based on the previously mentioned evaluation function. It works backwards from the position it wants to reach based on that position's score to the position where it initially is.

For a simple example: Suppose white can capture a knight (3 points), but sees that after that capture, black can take white's rook (5 points). The Minimax algorithm would prioritize preventing the white rook from being captured over capturing the black knight.

Minimax assumes both opponents play optimally where it assumes you play the best move and the opponent minimizes your advantage. By using a technique called Alpha-Beta Pruning, it eliminates branches that do not influence the final decision, which reduced the number of nodes evaluated.

In the earlier chess example, Alpha-Beta Pruning wouldn't evaluate all branches once it realises that white's rook being captured is disadvantageous. There would be no need to evaluate other moves if it would be guaranteed to end up worse for white anyways, so less time is spent on evaluating the less promising lines and divides its time to the more promising lines (see diagram below).



This method is used by chess engines when calculating a move to save calculation time and maximise the quality of each chess move by prioritizing certain features in a position. This also means that if all outcomes were to be calculated instantly by engines in the future, like Xilimbljor can, then minimax and alpha-beta pruning wouldn't be needed. It is also useful to consider that opening moves are studied very thoroughly and

therefore is easier to evaluate perfectly. Similarly, an endgame position with less pieces is also studied very thoroughly and therefore also easier for it to be perfectly evaluated, unlike a middle-game position that has never happened in chess before.

A limitation to using these evaluation techniques to compute, is that the engine cannot see far enough ahead and might suggest a suboptimal move which only becomes clear many moves later, this is called the horizon effect.

A way chess engines combat the horizon effect is by not relying on certain rules but rather playing many games against themselves to recognise patterns, and by using collected data to decide a move by using deterministic logic if two decisions are equally appropriate. For example, if an opening move as white has a 70%-win rate, and another move has a 65%-win rate, then the engine will likely prioritize playing the first move, even then, the best that chess engines can do is provide a very educated guess to what a good move is, and they will never perfect chess the way Xilimbljor has.

Although algorithms such as Minimax and Alpha-Beta pruning significantly reduce the complexity of the problem, they do not fully eliminate the uncertainty of chess. Due to the combinatorial explosion of possible positions, limitations in evaluation functions, and restricted search depth, chess remains fundamentally unsolvable beyond certain limits.

Turns out, even if both players play well, and games where players play obvious blunders are discarded, the number of games remains too vast. Even if you discarded 99.999% of all chess games, that still leaves 10^115 games to evaluate. Chess doesn't change when you play well, it remains infinitely complex.

In conclusion, chess, cannot be explained using brute force alone, as demonstrated by Shannon's analysis of the immense complexity of chess. While algorithms such as minimax and alpha-beta pruning reduce the number of positions evaluated, it still cannot evaluate the entire game in one go. Unlike Xilimbljor, they are reliant on heuristics, and for that reason has its limits with the horizon effect and cannot guarantee perfect chess. Although modern chess engines like AlphaZero and Stockfish play at an extremely high level, they only do so through approximation. Ultimately, chess remains a complex system where perfect foresight is computationally unobtainable in our universe, serving as a benchmark for future artificial intelligence models and problem-solving algorithms. Some might argue that technology will never reach that level, because Shannon's number isn't just big, it's astronomically beyond physical limits. Chess, then, is a strange kind of infinity: one that is entirely determined, and yet forever beyond our grasp. It reminds us that computation has limits, and that some questions, even when the rules are clear, may never have a single correct answer...

With the sole exception of Xilimbljor that is.

REFERENCES

Allis, L. V. (1994). *Searching for solutions in games and artificial intelligence* (Unpublished doctoral dissertation). University of Limburg. Retrieved from http://fragrieu.free.fr/SearchingForSolutions.pdf

Buttner, C. (2021). *High-level explanation*. ChessCoach. https://chrisbutner.github.io/ChessCoach/high-level-explanation.html

Desmos. (n.d.). *Chess TMR* [Desmos graph]. Retrieved April 8, 2026, from https://www.desmos.com/calculator/0pg7rnrgvv

Shannon, C. E. (1950). *XXII. Programming a Computer for Playing Chess* (Philosophical Magazine, Ser. 7, Vol. 41, No. 314). Bell Telephone Laboratories. https://vision.unipv.it/IA1/ProgrammingaComputerforPlayingChess.pdf